不用前缀和快速算出所有奇数长度子数组的和

——2021/8/29 Leetcode.1588 所有奇数长度子数组的和

- 题目：

> 给你一个正整数数组 `arr` ，请你计算所有可能的奇数长度子数组的和。
>
> 子数组 定义为原数组中的一个连续子序列。
>
> 请你返回 `arr` 中 所有奇数长度子数组的和。

- 示例：

```
输入：arr = [1,4,2,5,3]
输出：58
解释：所有奇数长度子数组和它们的和为：
[1] = 1
[4] = 4
[2] = 2
[5] = 5
[3] = 3
[1,4,2] = 7
[4,2,5] = 11
[2,5,3] = 10
[1,4,2,5,3] = 15
我们将所有值求和得到 1 + 4 + 2 + 5 + 3 + 7 + 11 + 10 + 15 = 58
```

- 思路：

一眼前缀和秒杀。但是前缀和的时间复杂度是O(n^2)，该题是可以达到O(n)的时间复杂度的。缩短时间的技巧在于，我们不必真的考虑每次都加上一个子数组的和，而是只**针对单独一个数字**考虑——探讨它的贡献度。何谓贡献度？即它出现在多少个数组中，例如本例中1数字出现在[1]、[1,4,2]、[1,4,2,5,3]三个数组中，贡献了三次，这三次的和为3。那么对每一个数字都用这种方法，最后加起来各自的贡献和，也是答案。

再次考虑每一个数字，**若要它位于一个奇数长度的数组中，那么它的左右两边挨着它的子数组的长度都必须同为奇或偶**。所以对于一个数字，我们可以考虑**当两边都为奇时，两边有多少个奇长度子数组供选择；两边都为偶时，又有多少个偶长度子数组供选择**。

再次举1数字为例。当两边取偶长度时，左边有[]（空集）一个子数组，右边有[]，[4,2]，[4,2,5,3]三个子数组，故偶情况下有1*3个贡献度，贡献度和为3；当两边取奇长度时，左边有零个子数组，右边有[1]，[1,4,2]，[1,4,2,5,3]三个子数组，故同理相乘贡献度为0，贡献度和为0。则总贡献度和为3+0 = 3；

用这种方法遍历每一个数字，即可达到O(n)的时间复杂度。

- 代码：

1.**前缀和**：

```c
int sumOddLengthSubarrays(int* arr, int arrSize)
{
    int sum[arrSize+1];
    sum[0] = 0;
```

```
    for(int i = 1;i <= arrSize;i++)
    {
        sum[i] = arr[i-1]+sum[i-1];        //预处理求前缀和
    }
    int ret = 0;
    for(int i = 1;i <= arrSize;i += 2)
    {
        for(int j = i;j <= arrSize;j++)
        {
            ret += sum[j]-sum[j-i];        //应用
        }
    }
    return ret;
}
```

2.**找规律**:

```
int sumOddLengthSubarrays(int* arr, int arrSize)
{
    int ret = 0;
    for(int i = 0;i < arrSize;i++)
    {
        int left = i + 1,right = arrSize - i;
        //当两边为偶数数组
        int left_even = (left+1) >> 1,right_even = (right+1) >> 1;
        //当两边为奇数数组
        int left_odd = left >> 1,right_odd = right >> 1;
        //求两边贡献度和
        ret += (left_even*right_even+left_odd*right_odd)*arr[i];
    }
    return ret;
}
```

- 后记:

感觉这个优化还蛮有用的，记下来吧。